Katrine Chow
CS 162 Winter 2018
Project 1 Design and Reflection

Project 1 – Design and Reflection Document

Problem:

Project 1 asks us to design and implement a version of Langton's Ant, a simulation of an ant on a board where the ant's movement is governed by two rules:

1) If on a white space, the ant turns 90 degrees right, turn the space black, and continues.
2) If on a black space, the ant turns 90 degrees left, turn the space white, and continues.

Requirements:

a) Ant class which contains information about the game board, the ant's location, and the ant's orientation.
b) User can specify size of board, the number of steps to take, and starting location of ant.
c) A menu that allows users to start or quit the Langton's Ant simulation. Once the game ends, the menu will prompt user to play again or quit.
d) Input validation required to make sure program does not crash from undesired input, and if incorrect input, repeatedly asks user until correct input is received.

Program Design:

See attached flowchart.

The program contains the Board class, the Menu class, the Ant class, and originally a Move class is planned.

The Board class would allocate and initialize a game board, and will keep track of board-related variables and functions.

The Menu class contains the menu. It is adaptable to other programs.

The Ant class contains all the variables keeping track of various characteristics of the ant (such as location and orientation). It also has functions to move the ant around the game board.

The Move class was intended to specialize in all movements required (such as turning, flipping around, flipping tile colors).

The goal was to create classes that can be used in other programs.

Test Plan:

See attached PDF.

Reflection:

This was quite an experience. Without the handholding from 161, I wasn't sure how to even get started. I began by sketching boards and how I imagine the program flow. I was hoping to write light and efficient code without too much hardcoding. Unfortunately I ended up doing a lot of hardcoding while I tried to debug the many issues I faced.

The worst of all issues was probably segmentation faults. I was not planning to create another dynamic 2D board to account for the tile colors, however, my initial plan of establishing a tile-only class/function was unsuccessful. Due to the lack of time I decided to create the second 2D array to keep track of the color of each tile. The board was initialized with Boolean true in each tile, representing white. The goal changed to having a second pointer set at the same tile location as the game board, and feed the Boolean value to the moveAnt function.

Having so many arrays and pointers probably caused the seg faults. I used gdb to help debug but I was unable to catch all of the irregularities by the deadline. I did however learned how to use *backtrace*, *frame*, and *where* commands and am now proficient in these tools!

The Menu was more time consuming than I had expected. In order to create a menu that can be used in other programs, I tried to use constants and get functions. This will need to be tweaked before use on future programs.

The Ant class went out of control! I find that I was adding variables and functions in order to accomplish what I wanted to do. I must admit that this was not planned out very well. There were many issues that came up that I did not predict, such as having to reset the ant's orientation in between shifting from turning right to turning left (and vice versa). I also didn't realize I could create a separate move function to take care of all the turning tasks, until after I've already hardcoded blocks of the same code. This is something I need to incorporate into my initial design for future assignments.

I was unable to implement the Move class as I had envisioned. I think somehow it made more sense to me to include it into the Ant class. However, I think if I was able to break this out into its own class (and file), it would make debugging much easier.

Overall, this was a very stressful but fruitful exercise. I feel like at some point I became desensitized to my code and had trouble stepping through the lines with a clear mind. This really showed me how valuable unit testing is. I think I would be able to troubleshoot the seg faults sooner if I had ironed out the kinks from unit testing.

Time to catch up on sleep. Despite everything, I think this is a great class opener. Am looking forward to future challenges.